Acceleration of Inductive Inference of Causal Diagrams

Olexandr S. Balabanov

Institute of Software Systems of NASU, Glushkov prosp.40, Kiev 03680, Ukraine

bas@isofts.kiev.ua

June 2011 (revised January 2013)

Abstract. *We propose a new principles and technique to speed up constraint-based algorithms for learning dependency structures from data. Novelty of proposed framework comes from implementing the rules of inductive inference acceleration, which can radically reduce a searching space for model's skeleton inference. The inductive inference acceleration rules facilitate fast identification of skeleton by performing such functions: recognition of edge presence (absence); recognition of some variables as non-members (obligate members) of supposed separator. It has been demonstrated that an algorithm, equipped with the proposed rules, can learn Bayesian networks (of moderate density) multiple times faster than well-known PC algorithm. Such improvement can be extended to the case of non-recursive graphical models, i.e. causal networks with hidden variables*.

Keywords

Inference of dependency networks from data, constraint-based algorithms, causal model, d-separation, conditional independence.

1 Introduction

Probabilistic modeling of dependency systems with graphical representation is attractive research topic on intersection of multivariate statistics, graph theory, information theory and artificial intelligence. Probabilistic graphical models are compact formalism for knowledge representation and efficient reasoning under uncertainty. Among all graphical models the most widespread are acyclic directed graph models (ADGs, also abbreviated as DAGs). An ADG model is comprehensible framework which is able to reflect causal relationship and provide efficient probabilistic inference from evidence [1–3]. ADG model class has appeared a powerful tool for numerous applications such as medical and technical diagnostics, gene expression analysis, robotics, speech recognition, classification, marketing, econometrics etc. Most commonly used are two kinds of ADG models: Bayesian networks (i.e. models with variables of nominal type) and Gaussian networks (i.e. linear models with continuous variables and normal disturbances).

We address the problem of improving the performance of constraint-based algorithms for inference of dependency networks from data. Learning a dependency networks from data is difficult and computationally expensive task due to the enormous size of the space of networks: the number of possible structures is super-exponential in the number of vertices (variables). We propose a new way to improve constraint-based algorithms by equipping the algorithms with special rules, called rules of inductive inference acceleration. This rules are strongly justified by acyclic property of digraph, the criterion of d-separation and appropriate version of the Causal faithfulness assumption. The rules of inductive inference acceleration can radically reduce a run-time of structural inference via cutting branches of search for separators. Experimental evaluation demonstrates that our algorithm ('Razor') performs learning Bayesian networks of moderate density multiple times faster than well-known PC algorithm. The error rate of 'Razor' algorithm is just insignificantly worse than that of PC algorithm.

2 Basic Definitions

Taking into account a one-to-one correspondence between variables of a model and vertices of corresponding graph we shall use terms *variable* and *vertex* interchangeably. If there is an arc (directed edge) $X \to Y$ in a graph, then vertex (variable) X is said to be a 'parent' of vertex Y. When ignoring the directions of arc $X \to Y$, we will call it an edge and denote $X — Y$. Vertices connected by arc (edge) are said to be adjacent. We denote by $\mathrm{F}(X)$ a set of parents of vertex X,

by Adj(X) – a set of vertices adjacent to X. A path is a sequence of incident edges X—··—Y without repetition of intermediate vertices. A strictly oriented path (dirpath) is a path $X\to\cdots\to Y$ on which all edges are oriented toward the same end of the path. When ignoring the directions of all arcs in a graph G, we get the skeleton of graph G. Acyclic directed graph (ADG) is digraph which contains no one cyclone (dircycle) $X\to\cdots\to X$.

An ADG model M is defined as pair (G, J), with G being acyclic directed graph, and J being attributed parameters. A parameters of ADG-model (Bayesian network in part) are defined as $p(X|\mathrm{F}(X))$. It worth to note that a parameters of Gaussian network are more conviniently defined as coefficients of linear equations (attributed to edges) and dispersions of variable values.

The Markov property of ADG model is formalized through the criterion of d-separation [1–3]. This criterion is defined purely in graphical terms and can be efficiently applied. If a set $\mathbf{Z}$ of vertices d-separates vertices X and Y, then $\mathbf{Z}$ is said to be a separator for pair $X,Y\notin\mathbf{Z}$, denoting this by Ds(X; $\mathbf{Z}$; Y). For opposite fact (when d-separation is not met) we will use denotation ~Ds(X; $\mathbf{Z}$; Y).

An assertion of conditional independence of variables X and Y given a set of variables $\mathbf{Z}$ will be denoted by predicate Pr(X; $\mathbf{Z}$; Y), where $X,Y\notin\mathbf{Z}$. This independence means that $p(XY|\mathbf{Z})= p(X|\mathbf{Z})\cdot p(Y|\mathbf{Z})$. In Gaussian networks a similar assertion is usually expressed by zero value of partial correlation coefficient. Unconditional independence is a special case of conditional independence with empty condition, that is Pr(X; {}; Y), or Pr(X;; Y) for short. If Pr(X; $\mathbf{Z}$; Y), holds we call $\mathbf{Z}$ an empirical separator for (X, Y).

It is known [1–3], that the fact of d-separation in G entails corresponding probabilistic conditional independence in M= (G, J):

$$\mathrm{Ds}(X;\mathbf{Z};Y)\Rightarrow\ \mathrm{Pr}(X;\mathbf{Z};Y).$$

Thus all Markov conditional independencies can be read off the model's graph regardless of parameter values. Apparently, there no one d-separator exists for adjacent vertices in a ADG. Class of ADG models naturally generalizes to more expressive class of causal networks which allow reflecting a presence of latent variables.

3 The Problem and Theoretical Justification for Improving Model Induction

An important problem is inferring the structure of dependency networks from data (this process is sometimes called causal discovery). Model inferring can provide insights into the underlying data generation process. Two major classes of methods exist for learning the structure of causal networks. One class comprises so-called score-based methods, which learn the structure by conducting a search in the space of all structures in an attempt to find the structure of maximum score. The score metric is usually penalized log-likelihood, for example, the Bayesian Information Criterion (BIC), or a posteriori probability etc. **A** second class of algorithms works by exploiting the fact that a causal network implies the existence of a set of conditional independence statements between sets of domain variables. These algorithms (called constraint-based or independence-based) use the outcomes of a number of conditional independence tests to infer a consistent structure. (It is possible to identify a structure up to equivalence class only). In this paper we address some open problems related to the constraint-based algorithms and provide useful improvement.

Constraint-based algorithms for learning the model's structure from data rely on quite heavy assumptions, such as the Causal faithfulness assumption and the correctness of independence tests. The Causal faithfulness assumption [2–5] may be expressed by the rule:

$$\forall X,Y\notin\mathbf{Z}:\ \mathrm{Pr}(X;\mathbf{Z};Y)\Rightarrow\mathrm{Ds}(X;\mathbf{Z};Y). \qquad (1)$$

Taking into account that reliability of conditional independence tests is sensitive to sample bias, accepting the faithfulness assumption practically may lead to errors. A constraint-based algorithm deletes an edge X—Y when finds a fact that variables X and Y are conditionally independent under some condition $\mathbf{Z}$. So a constraint-based algorithm tries to find a separator for each pair (X, Y). The smaller data sample is the less reliable inference turns out to be. To justify some particular inference algorithm, it may be sufficient to accept a more robust version of the Causal faithfulness assumption than (1).

Constraint-based algorithms are more fast, than 'search&score' algorithms. But both groups of algorithms turn out to be unsatisfactory for many practical tasks. When data are discrete and number of variables goes beyond a few tens, the algorithms require overly many runtime (although for Gaussian networks the execution time may be acceptable for more than hundred variables). This limitation comes from combinatorial nature of the learning algorithms, since in essence the algorithm searches for separator for every pair of variables. This search is especially hard when causal ordering of variables is not given (this is below assumed to be the case). The main goal of our proposals is to increase a speed of inductive inference of ADG models.

The most popular constraint-based algorithm seems to be the PC-algorithm [2–5]. It performs inference in three phases: 1) inference of graph's skeleton; 2) orienting edges of graph inferred; 3) calculating model parameters. The first phase is point of our attention. PC-algorithm starts from a complete, undirected graph and deletes recursively edges having obtained independence facts. The algorithm iteratively forms tentative separators in growing order of cardinality. PC-algorithm runs in the worst case in exponential time (as a function of the number of nodes), but if the true ADG is sparse, a runtime appears to be acceptable (polynomial).

Searching for separators is a core task of model inference. It is known to be a hard problem. The key invention of PC-algorithm is the following principle: to include in a tentative separator for pair (X, Y) only those variables which are supposedly adjacent to X or to Y. This considerably reduces a search space and renders the inference much faster. But for networks of even moderate density searching for separators still remains very complex and expensive. A separator $\mathbf{Z}$ may be a set of any cardinality (of course, no larger, than (n-2), with n being a number of variables). Thus the problems with such methods are combinatorial complexity and low reliability in edge identification. This comes from unreliability of result of independence test under small data sample. When a cardinality of condition $\mathbf{Z}$ grows up (in discrete model), a data is splitting (and sample degrades). For reliable recovery of a 'true' model it is desirable to come with tests of low rank whenever possible. Perhaps the worst consequence of PC-algorithm strategy is typical situation when there edge X—Y exists, but PC continues an attempts to find a separator for (X, Y) and performs superfluous non-productive work. Indeed, for each pair of adjacent variables (X, Y) the algorithm examines all subsets of Adj(X) and all subsets of Adj(Y) as tentative separators. Therefore, of especial impotance is to recognize the edge presence as early as possible.

Our main purpose is further to focus searching for separators and to render an algorithm more clever and efficient. The key idea to achieve the goal came from perceiving that there exist some implications among conditional independence facts in ADG model. Than is, when certain d-separations are satisfied in digraph G, this immediately constraints other d-separations (with overlapping subsets of vertices) in G. Moreover, when a model meets certain list of conditional independencies, this implies some other certain conditional independencies (or renders some independencies prohibited). So, having obtained a pattern of dependencies/independencies, we can constraint space of possible separators or even identify edges. Understandably, (in)dependencies of low rank should be used to optimize searching for independencies of higher rank. Knowing of some minimal separators assists to narrow searching for "contiguous" separators (in neighborhood).

A starting point for elaboration and justification a new inference rules and principles is notion of locally-minimal separator [6].

Definition 1. A d-separator $\mathbf{Z}$ for pair (X, Y) is said to be locally-minimal separator, iff for any $W \in \mathbf{Z}$ it is satisfied $\sim$Ds(X; $\mathbf{Z}\backslash\{W\}$; Y). In words, removing any member of locally-minimal d-separator from the separator destroys d-separation at hand.

For ADG this definition is known to be equivalent to the following one. A set $\mathbf{Z}$ is called a locally-minimal d-separator for pair (X, Y), iff Ds(X; $\mathbf{Z}$; Y) and $\sim$Ds(X; $\mathbf{Z'}$; Y) for any $\mathbf{Z} \subset \mathbf{Z'}$ ($\mathbf{Z} \neq \mathbf{Z'}$).

A d-separator $\mathbf{Z^*}$ for pair (X, Y) is said to be minimal in G, iff there is no one d-separator $\mathbf{Z}$ for pair (X, Y) in G such that $|\mathbf{Z}| < |\mathbf{Z^*}|$. In words, minimal separator is the one of minimal cardinality.

As demonstrated in [6, 7], any member of some minimal (locally-minimal) separator must meet a certain necessary conditions. This was formalized via appropriate statements and rules. The rules and statements are derived from the criterion of d-separation and acyclic property of digraph. Developed rules assist to rule out 'false' (or unnecessary) candidates to a (locally)minimal separator. These rules apply patterns of (in)dependencies of zero and one rank only. The proposed rules facilitate fast identification of edges (or its absence) in a dependency structure.

All proposed (and others supposedly useful) rules may be classified (grouped) accordingly to the roles (functions) they perform. There are the following four roles of rules developed: 1) recognizing of edge presence; 2) recognizing of edge absence; 3) deleting some variables from candidates in d-separator; 4) recognizing of some variables as obligate members of supposed separator. It is useful to involve a fifth family of rules, which recognize some variables as unnecessary for separation (even though a minimal separator may be missed).

One of the most important rules is the following one.

Rule of 'placing aside'. If there Ds(Z; X; Y) & $\sim$Ds(Z;; Y) holds in G, then vertex Z is not a member of any locally-minimal d-separator for pair (X, Y) in G.

The following rule can play important role because it recognizes edge presence, thus finishing a search for separator for respective pair.

'Lack of separator's pivot' rule. If $\sim$Ds(X;; Y) and there exists no one vertex Z which satisfy $\sim$Ds(Z;;X) & $\sim$Ds(Z;;Y) & $\sim$Ds(Z; X; Y) & $\sim$Ds(Z; Y; X) in G, then edge X—Y is present in G.

One can easily verify that it is sufficient to employ the two presented above rules to provide ability for identifying any structure in a class of forest or poly-forest (poly-tree) – subclasses of ADG – by test of zero- and first-rank only [8]. An example of forest (tree) is depicted in figure 1a (such a structure is widely known as Naпve Bayes classifier). A structure depicted in figure 1b exemplifies a general case of ADG model.

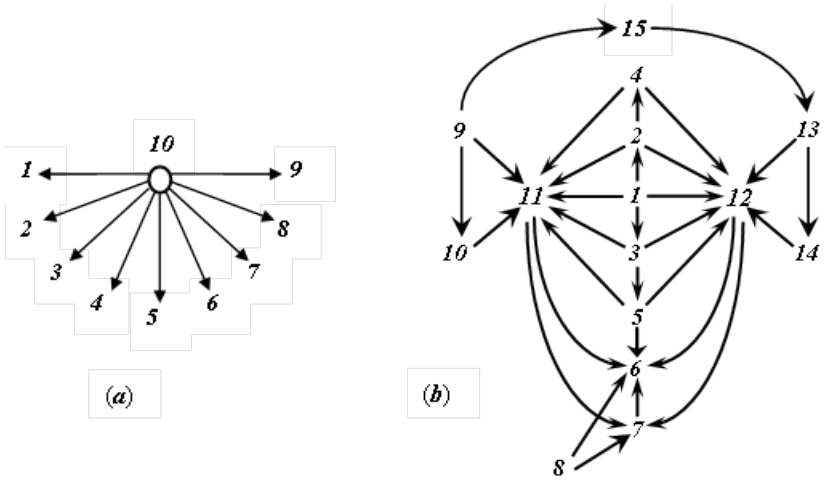

Fig. 1. Two examples for illustration: (a) tree; (b) ADG model.

Also very useful may be the following rule.

Rule of 'alien gene'. If for some Z there exists W in a digraph G such that ~Ds(W;;Z) & ~Ds(Z;;X) & ~Ds(Z;;Y) & Ds(W;;X) & Ds(W;;Y) holds in G, then vertex Z is not a member of any locally-minimal d-separator for pair (X, Y) in G.

One can find more rules in [6, 7].

To obtain rules of inductive inference acceleration (which may be utilized for inference from data) it is needed to replace a graphical predicates in the rules mentioned above by isomorphic empirical predicates. Such conversion is principally justified by the Causal faithfulness assumption. Appropriate versions of the Causal faithfulness assumption (sufficient to justify inference of model's skeleton from statistical data) are discussed in [6, 7, 9]. It is worth to note that empirical versions of the rules are (should be) somewhat narrowed to achive more efficiency and reliability. Then we can employ empirical counterparts of mentioned rules in induction algorithm. Of course, applying empirical rules of acceleration for model induction from real data sample brings additional risk of mistakes. Such a risk of mistakes should be evaluated experimentally.

4 Experimental Evaluation of Proposed Improvements for Model Induction

Using a known PC algorithm as starting point and base, we have developed a new constraint-based algorithm for inference of dependency structures from data. New series of algorithms (which we call 'Razor') are augmented with the rules of inductive inference acceleration, proposed above. We have decided to retain the following main principle of PC algorithm. Only vertices adjacent to X or to Y are used as candidates to tentative separator for (X, Y). When implementing the first version of our algorithm ('Razor-1.1'), we selected several acceleration rules, taking into account their expected efficiency and reliability. For more detailed description of Razor-1.1 algorithm see [9]. All compared algorithms (including PC and Razor-1.1) were implemented in MATLAB package obeying the same style. Evaluation and comparison of the algorithms were performed at two levels (in two modes): graphical and empirical mode. Graphical mode ('logical simulation') means, that d-separation facts are given in the input of algorithm. Under empirical mode the results of independence tests on real data sample are used. Results of logical simulation on collection of examples (models) have confirmed the correctness of developed algorithm.

As noted above, it is sufficient to enrich an algorithm like PC with just the two rules (placing aside rule and "lack of separator's pivot" rule) for enabling algorithm to recover a model, whose structure falls into a class of forest or poly-forest, by executing tests of unconditional independencies and conditional independencies of first rank. In particular, algorithm Razor-1.1 would identify a structure of figure 1a by tests of 0-rank and 1-rank only, like specialized

algorithms. In contrast, basic PC algorithm when inferring the same model would work out test up to 8th rank (inclusively).

An efficiency of our algorithm 'Razor-1.1' in general case of ADG model may be illustrated by the following example. Chosen model structure is depicted in figure 1b. It consists of 15 vertices and 30 edges. To reconstruct a skeleton of this model (in logical simulation mode), algorithm PC has required about ten thousand of tests of d-separation, including tests of rank 8. In contrast, algorithm Razor-1.1 has completed reconstruction using tests up to rank 4 only. (Number of tests was reduced approximately by factor ten.)

To accomplish more realistic examination of developed algorithm, we perform a series of inductive inference from real data samples. A wide collection of synthetic models were used for experiments. Model structures were generated randomly, all with 20 vertices (variables). Number of edges ranges from 40 to 70. A model's parameters were also generated randomly for each structure, with variables being binary and ternary. Then data samples (of sizes 10000 and 20000) were generated from the models. Results for samples of size 20000 are presented below.

Absolutely all accomplished experiments demonstrated better performance (speed) of Razor-1.1 algorithm relative to that of PC. Also Razor-1.1 tends to reduce a maximum rank of tests executed. But Razor-1.1 regularly performs worse in accuracy (it commits more mistaken edges). Nevertheless, a number of edge loses (omissions) doesn't increase, a number of edge additions increases insignificantly, whereas acceleration grows radically. In part, for the group of models with 20 vertices and 60 edges algorithm Razor-1.1 performed inference by 5.8 times faster than PC (in average), while number mistakes increased by 1.4 times. Results obtained for the group of models with 20 vertices and 50 edges are presented in figure 2. There are depicted runtime values of both algorithms. Acceleration values ranges from 2.6 to 15 times. Note, that average rate of mistakes was 19% for PC and 22% for Razor-1.1.

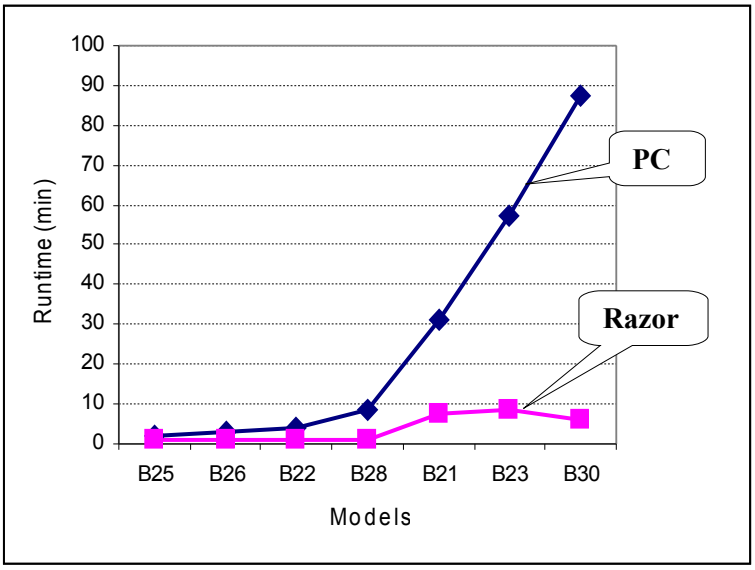

Fig. 2. Performance of two examined algorithms.

Especially interesting is the fact, that acceleration appears higher for more complex inferences. For instance, to infer the model labeled as B30, our algorithm spent 6 minutes only, while algorithm PC spent about 1.5 hour. For the model labeled as B38 (with 60 edges), corresponding runtimes appeared to be 13 minutes versus 2 hour 40 minutes. So, proposed methodology provides considerable improvement of inductive inference. It should be noted, that such high values of runtime are caused by the interpretation mode of program execution. But now we are interested in relative run-time only.

5 Conclusion

We have developed new principled tools to enforce constraint-based algorithms for learning dependency structures and causal networks from data. Novelty of our proposals comes from implementing the rules of inductive inference acceleration, which can radically reduce a search space for skeleton inference, thus reducing computational complexity. The rules for efficient picking up a minimal d-separating sets in an ADG models are deduced from the criterion of d-

separation and acyclic property of digraph. The innovation proposed allows overcoming one of the main shortcomings of known methods and algorithms for causal inference – overly combinatorial complexity. The goal is achived by equipping the well-known PC algorithm with rules of inductive inference acceleration. This provides an important ability to recognize edge presence or edge absence, and also to recognize some variables as unnessesary or as obligate in searching for supposed separator. In part, modification proposed facilitates recovering a dependency forest or poly-forest by means of first-rank tests only (while algorithm still retains correctness in general case). Experiments have demonstrated that algorithm equipped with the proposed rules performs learning Bayesian nets (of moderate density) multiple times faster than PC algorithm. At the same time, number of errors grows much more slowly. Thus the inductive inference acceleration rules facilitate fast identification of skeleton of causal model.

It is worth to note, that most of the rules of inductive inference acceleration may be extended to the case of causal networks with latent variables [10] (some corrections to the algorithm should be done).

References

1. Pearl J. CAUSALITY: models, reasoning, and inference. – Cambridge Univ. Press N.Y., 2000. – 384 p.
2. Spirtes P., C. Glymour, and R. Scheines. Causation, prediction and search. (2nd Ed.), New York: MIT Press, 2001. – 543 p.
3. Neapolitan R.E. Learning Bayesian networks. – Upper Saddle River: Prentice Hall, NJ, 2004. – 693p.
4. The TETRAD Project: Constraint based aids to causal model specification / R. Scheines, P. Spirtes, C.Glymour et al. // *Multivariate Behavioral Research*. – (1998). – Vol. 33. – No 1. – P. 65–118.
5. Ramsey J., P. Spirtes, and J. Zhang. Adjacency-faithfulness and conservative causal inference / Proceedings of the 22nd Conf. on Uncertainty in Artificial Intelligence. – Oregon, AUAI Press, 2006. – P. 401–408.
6. Balabanov A.S. Minimal separators in dependency structures: Properties and identification // *Cybernetics and Systems Analysis*. – (2008). – Vol. 44, – No 6. – P.803–815. – Springer N.Y.
7. Balabanov A. S. Construction of minimal d-separators in a dependency system // *Cybernetics and Systems Analysis*. – (2009). – Vol. 45. – No 5. – P. 703–713.
8. Balabanov O. S. Accelerating algorithms for Bayesian network recovery. Adaptation to structures without cycles (in Ukrainian) // *Problems in programming journal*. – (2011). – No 1. – P.63–69. – Kiev, Ukraine, ISBN 1727-4907.
9. Fast algorithm for learning the Bayesian networks from data / Alexander S. Balabanov, Alexander S. Gapyeyev, Anatoliy M. Gupal, Sergey S. Rzhepetskiy // *Journal of Automation and Information Sciences*. – (2011). – Vol. 43. – Issue 10. – P. 1-9.
10. Balabanov A. S. Logic of minimal separation in causal networks // *Cybernetics and Systems Analysis*. – (2013). – Vol. 49. – (to appear).